Ficha 9 – Vetores de Ataque

HTTPS, *sniffing* e principais vetores de ataque dos sistemas digitais

Tópicos abordados:

- *Sniffing*;
- HTTPS (TLS)
- *SQL Injection*
- *Replay Attacks*
- XSS (*Cross-Site Scripting*)

©2017: {rui.ferreira, ricardo.p.gomes, nuno.reis}@ipleiria.pt

1. Objetivo e projetos disponibilizados

O objetivo desta ficha (9) é testar a implementação das fichas anteriores (7 e 8), verificando os mais comuns vetores de ataque a sistemas digitais. Como se pode observar na Figura 1, a seleção representa o conjunto de aplicações que vai ser utilizado nesta ficha:

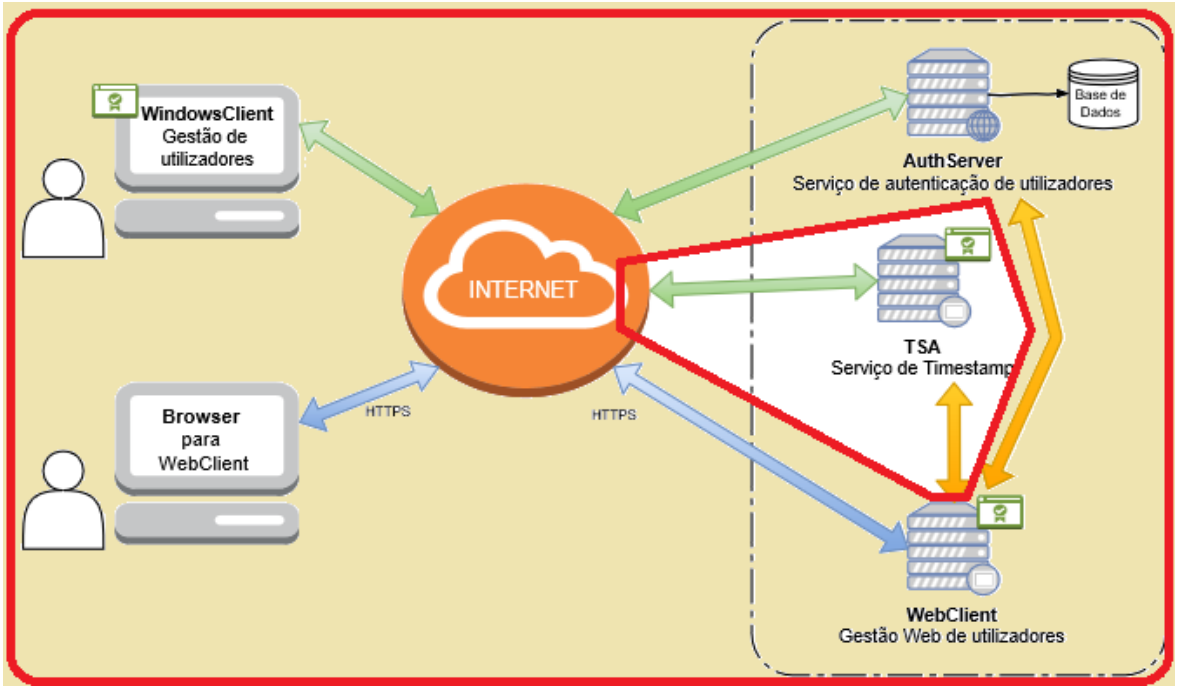

Figura 1 - Aplicações a implementar/atualizar nesta ficha

Para continuar a implementação do sistema vamos conhecer qual novo código fornecido e como os pode juntar aos projetos existentes. A estrutura final da *Solution* deve ser a seguinte:

Figura 2 – Estrutura da *Solution* para esta ficha

Como se pode verificar, na *Solution* "ei.si-worksheet7_8_9" devem estar presentes quatro projetos:

- AuthService – Serviço Web de autenticação e gestão de utilizadores;
- TSA – Serviço Web de *Timestamping*;
- WindowsClient – Aplicação Windows para utilização dos serviços anteriores;
- Website – Aplicação ASP.Net.

Destes apenas se vai apresentar o novo projeto e também as novidades adicionadas aos projetos existentes.

1.1. Website

O projeto "Website" é uma aplicação Web e tem a seguinte estrutura:

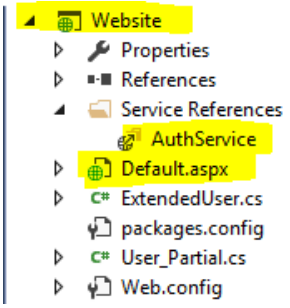

Figura 3 - Estrutura do projeto TSA

De salientar os seguintes ficheiros:

- Default.aspx e Default.aspx.cs – interface e respetiva implementação das funcionalidades a disponibilizar pelo website;
- AuthService – referência para o serviço de gestão e autenticação de utilizadores.

1.2. WindowsClient

Quando a novidades no "FormMain" há apenas uma a salientar (no retângulo a vermelho).

Os elementos principais para utilizar são:

- Button - btnReplayAttack

1. Abra a *Solution* "ei.si-worksheet7_8_9" e:
    a) Verifique que os projetos mostrados anteriormente estão presentes;
    b) Execute o projeto "WindowsClient" e verifique que todas as funcionalidades estão a trabalhar corretamente.

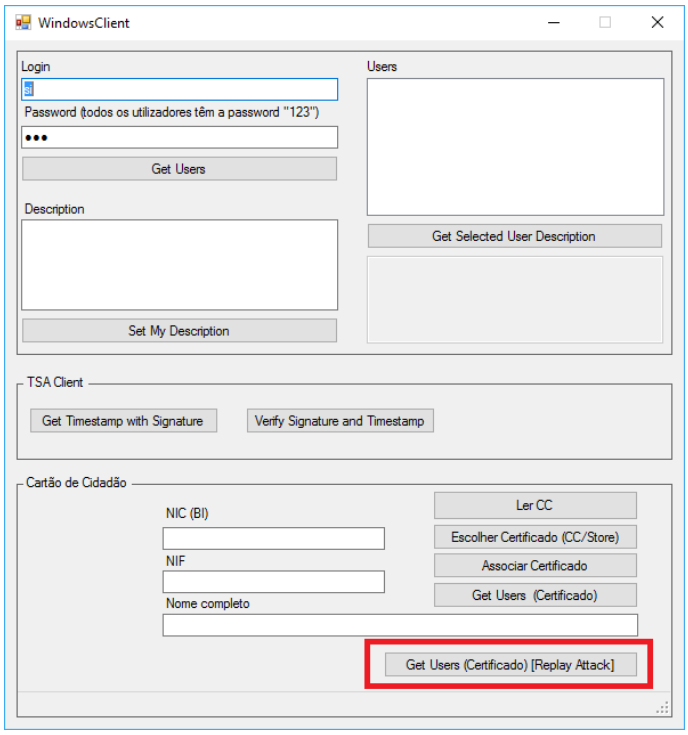

Figura 4 - Formulário "FormMain" e novos componentes

2. Vetores de Ataque (parte 1)

Neste capítulo apenas irá ser iniciada a abordagem a dois dos vários vetores existentes: SQL *Injection* e *Replay Attack*. O motivo é mostrar que a utilização de um canal de comunicação não seguro permite com que um maior número de ataques aconteça, mas que a utilização de um canal seguro também não previne todos os tipos de ataques (como é o exemplo do SQL *Injection*). Deste modo, é necessário ter em consideração os principais aspetos para proteger as aplicações desses ataques.

2.1. SQL Injection

O ataque por SQL *Injection* pode ser visto como o manipular do código SQL antes de este ser executado no motor de base de dados. Isto é conseguido porque o atacante consegue injetar uma *string* maliciosa através de uma qualquer interface e também porque o código da aplicação permite aceitar essa manipulação.

Neste ponto vamos apresentar o conceito, mostrando um exemplo simples e outros mais complexos, mas cujo objetivo é sempre o mesmo: alterar o normal funcionamento da aplicação.

No Capítulo 5 volta-se a falar neste tipo de ataque para mostrar a técnica mais comum para o evitar.

1. No projeto “WindowsClient”:

    a) Verifique se consegue obter a lista de utilizadores com o utilizador/password “si/123”;

    b) Verifique se consegue obter a lista de utilizadores com o utilizador/password “admin/123”;

    c) Efetue um “ataque por SQL *Injection*” nas caixas de texto “Utilizador” e/ou “Password” e verifique se consegue obter a lista de utilizadores usando um utilizador/password inexistentes;

    d) **[extra]** Use o botão “Set My Description” de modo a colocar todos os utilizadores com o campo *Description* a “SQLinjection”.

2. Analise o código dos projetos envolvidos no problema apresentado de modo a perceber como foi possível o ataque.

2.2. Replay Attack

Outra forma de ataque é ao “*replay attack*”, que consiste no envio de pedidos, desfasados no tempo, com o intuito de fazer a mesma ação do pedido original. Como exemplo: se conseguirmos obter um pedido de autenticação e o utilizarmos num futuro próximo então estamos a fazer-nos passar por outra pessoa.

Neste ponto vamos simular um ataque por repetição e também apresentar uma solução de como resolver este problema para as aplicações.

1. **[extra]** No projeto “WindowsClient”:

    a) Para efetuar um “*replay attack*” vamos simular a captura de um pacote de dados na rede de uma forma simples: colocar um *sniffer* no método do botão “Get Users (Certificado)”:

    ```
    byte[] pkcs7Signature = signedCms.Encode();
    this.dataForReplayAttack = pkcs7Signature;
    ```

    Nota: `this.dataForReplayAttack` é um *array* de bytes e é um atributo da classe;

    b) Use o botão “Get Users (Certificado)” e escolha um certificado que lhe permita obter a lista utilizadores;

    c) Analise e verifique se o código do botão “Get Users (Certificado) [Replay Attack]” está sem erros.

    Nota: este código vai utilizar o valor capturado na escuta anterior;

d) Carregue no botão “Get Users (Certificado) [Replay Attack]” e verifique se o ataque foi efetuado com sucesso;

e) Para se iniciar o processo de prevenção conta este tipo de ataque, garanta que o processo de *login* por certificado digital envia sempre um *timestamp* na assinatura digital (fornecido, para já, pelo relógio do computador onde está a ser executado o “WindowsClient”);

f) **[extra]** Altere o processo elaborado na alínea anterior para que a informação a enviar ao serviço seja proveniente de uma TSA.

2. **[extra]** No projeto “AuthServer”:
   a) Na base de dados, adicione um campo à tabela “Users” chamado “Timestamp”;
   b) Atualize o método “GetUsersByCertificate” com a autenticação por certificado digital de modo a prevenir *Replay Attacks*;

3. **[extra]** No projeto “WindowsClient”:
   a) Execute e teste as operações do ponto **2.2.1.** alíneas **a)** e **d)**;

3. Análise do Tráfego da Rede

Para verificar que existe tráfego na rede que não garante a confidencialidade da informação vamos usar software que permite estar à escuta (*sniffing*) de dados que não deveriam ser de domínio público. O software escolhido é o Microsoft Message Analyzer, que permite fazer uso da interface de *loopback* (*localhost*/127.0.0.1) e, deste modo, otimizar procedimentos que de outra forma demorariam algum tempo até que todo o cenário de testes estivesse implementado.

1. Faça *download* e instale aa aplicação “Microsoft Message Analyzer” (última versão disponível) e:
   a) Crie uma nova sessão: “sniffing-SI”, com a *data source*: “Live Trace” e *Selected Scenario*: “Local Loopback Network”, configure o *Provider* para aceitar todo o tipo de ligações IPv4 e IPv6 e inicie uma captura;
   b) Pare a captura e faça uma análise do que foi capturado: origem, destino, módulos (protocolos) e descrição;

3.1. Análise de tráfego HTTP

1. Execute a *Solution* "ei.si-worksheet7_8_9".

2. Inicie mais uma captura de pacotes da rede, com as mesmas definições utilizadas no ponto anterior.

3. Utilize o formulário do "WindowsClient" e utilize o botão "GetUsers" com qualquer utilizador/password.

4. Pare a captura e faça uma análise dos pacotes mostrados para tentar verificar se é possível obter a informação submetida anteriormente.
   Nota: utilize filtros para minimizar o tempo de pesquisa.

5. Que pode concluir com as ações realizadas?

4. HTTPS

De modo a garantir segurança no acesso aos websites disponibilizados pelos servidores Web é necessário configurar o serviço Web de modo a que este estabeleça um canal de comunicação seguro com os clientes (*browsers*). Neste momento, em modo de desenvolvimento, a maneira mais simples é configurar o IIS Express utilizando o Visual Studio.

1. No projeto "AuthService":
   a) No "Solution Explorer" escolha o projeto, faça "F4" e ative a opção "SSL Enabled";

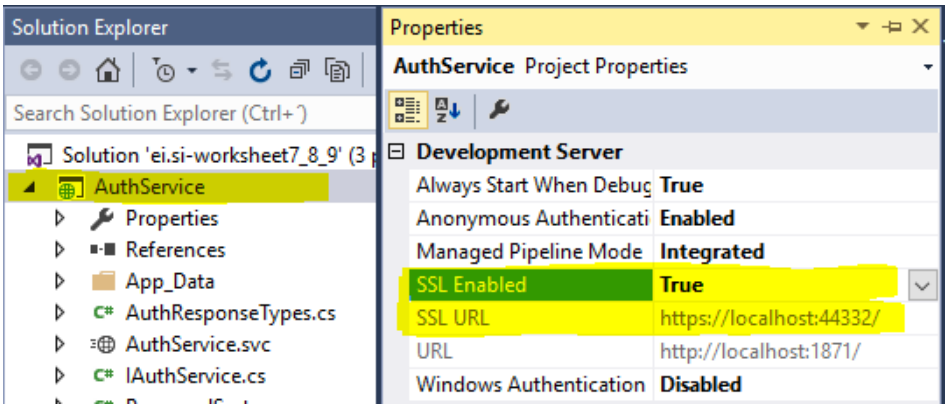

Figura 5 - Implementação de HTTPS no Visual Studio

   b) Faça "Rebuid" ao projeto e garanta que estas atualizações ficam disponível no IIS Express, como apresentado na próxima figura:

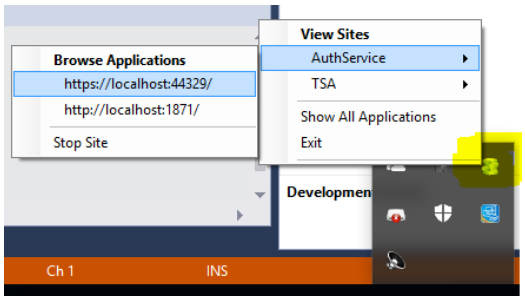

Figura 6 - Confirmação que o IIS Express está a disponibilizar o serviço por HTTPS

2. No projeto "WindowsClient":
    a) Atualize a referência para o serviço "AuthService" de modo a utilizar o novo meio de comunicação seguro (HTTPS);
    b) Teste o programa para verificar que todas as funcionalidades continuam disponíveis. Nota: se obtiver um erro por mais que um "*endpoint*", edite o ficheiro "app.config" e comente o "*endpoint*" que faz a ligação ao serviço "AuthService" via protocolo HTTP.

3. Para "provar" que existe efetivamente um canal seguro entre cliente e servidor, volte a executar todos os passos do ponto 3.1: Análise de tráfego HTTP.

5. Vetores de Ataque (parte 2)

Os próximos exercícios pretendem mostrar como se pode utilizar alguns dos principais vetores de ataque a aplicações digitais, bem como, de seguida, proteger as aplicações contra os mesmos.

5.1. SQL Injection

O ataque por SQL *Injection* volta a estar aqui presente para, finalmente, mostrar a técnica mais comum para o evitar: *SQL Parameters*. É, assim, importante ter presente este conceito e saber como o usar de forma adequada.

1. No projeto "AuthServer":
    a) Atualize o código que permite evitar o ataque por SQL *Injection* à funcionalidade "GetUsers";

    Nota: SQL *Parameters* é uma das técnicas mais conhecidas, simples e eficazes;

```
cmd.CommandText = "SELECT * FROM Users WHERE login = @login";
cmd.Parameters.Add(new SqlParameter("login", login));
// ou
cmd.Parameters.AddWithValue("login", login);
```

2. No projeto "WindowsClient":
    a) Execute e teste as operações do ponto **2.1.1.** alíneas **b)** e **c)**;

3. **[extra]** Atualize o projeto "AuthServer" de modo a que fique completamente protegido contra ataques por SQL *Injection*.

5.2. Cross-site scripting (XSS)

Este tipo de ataque (XSS) tem o mesmo conceito do SQL *Injection*, onde a diferença principal é o tipo de código que se "injeta": tipicamente *javascript*. A ideia é utilizar a interface disponibilizada pela aplicação (normalmente Web) para escrever texto "especial" que depois vai ser executado por qualquer utilizador que use a aplicação. Com este tipo de ataque pode-se obter informação de outros utilizadores, sendo exemplo disso o roubo do identificador de sessão (*Session Hijackin*).
Os próximos exercícios pretendem mostrar um exemplo deste ataque e uma maneira de proteger do mesmo.

1. Adicione o projeto "Website" à *Solution*, como mostrado no capítulo 1:
    a) Analise o código do ficheiro principal "Default.aspx.cs" e ficheiros auxiliares: "ExtendedUser.cs" e "User_Partial.cs";

2. No projeto "Website":
    a) Analise agora com mais atenção o código dos ficheiros "Default.aspx" e "Default.aspx.cs", para tentar prever o que vai acontecer quando este for executado;
    b) Inicie a aplicação e verifique qual o resultado da execução do código anterior.

3. No projeto "WindowsClient":
    a) Use o botão "Set My Description" e garanta que a descrição do utilizador "si" é:

    ```
    User X - <script>alert("Aqui podia estar a obter dados e enviar para alguém");</script>
    ```

    Nota: se preferir, pode fazer isto diretamente na base de dados.

4. No projeto "Website":
    a) Inicie novamente a aplicação e verifique qual é o problema existente;
    b) Altere a implementação deste projeto de modo a garantir que ataques de *XSS* não são efetuados com sucesso;

Nota: este mecanismo de proteção deveria ser efetuado no *input* dos dados, estando a ser feito neste momento (*output*) apenas por motivos de otimização do tempo da aula;

c) Inicie outra vez a aplicação e verifique que efetuou corretamente as alterações anteriores.